Modeling a spacewire architecture using timed automata to compute worst-case end-to-end delays

Jérôme Ermont, Christian Fraboul

Modeling a Spacewire architecture using Timed Automata to compute worst-case end-to-end delays

Jérôme Ermont, Christian Fraboul
Université de Toulouse - IRIT - INPT/ENSEEIHT, 2 rue Camichel, 31000 Toulouse, France
{jerome.ermont, christian.fraboul}@enseeiht.fr

Abstract

Spacewire is a real-time communication network for use onboard satellites. It has been designed to transmit both payload and control/command data. To guarantee that communications respect the real-time constraints, designers use tools to compute the worst-case end-to-end delays. Among these tools, recursive flow analysis and Network Calculus approaches have been studied. This paper proposes to use the model-checking approach based on timed automata. A case study based on an industrial one is shown. Our approach is compared with recursive flow analysis and Network Calculus.

Keywords. *Timed automata, UPPAAL modeling, Spacewire network, Worst-case delays analysis*

1 Introduction

SpaceWire [9] is a communication network for use onboard satellites which has been developed by the European Space Agency and the University of Dundee. It provides high-speed data exchanges, between sensors, memories, processing units and downlink telemetry.

One goal of SpaceWire is to carry both the payload and the command/control traffic instead of using dedicated buses, as MIL-STD-1553 buses, for both of them. Different requirements are needed: low throughput and very strict time constraints for command/control traffic and a sustained high bandwidth for payload.

SpaceWire is based on a part of the IEEE-1355 standard [1] and uses packet switching to connect several equipments. Due to the space requirements (specially the radiation tolerance), a minimal amount of data can be stored in the routers. To ensure this, SpaceWire uses wormhole routing: packets are not stored completely but can be forwarded as soon as the output port is free. If the output port is not free, the packet is blocked. In that case, the packet cannot be transferred from the upstream router blocking other packets. The consequence is a variation of the end-to-end (ETE) delays for the packets. A method to verify that the time constraints are guaranteed must be defined.

A similar problem arises in the context of avionics where an upper bound has to be computed in respect to the certification. Two solutions are based on Network Calculus [5] and Trajectories [4]. However, the obtained upper bounds are pessimistic due to their assumptions. Other works have been devoted to compute the exact ETE delays of such networks. Existing model checking approaches [5, 8] implement an exhaustive analysis of all the possible scenarios. However, it cannot be applied to Avionics Full DupleX switched Ethernet (AFDX) configurations with more than 10 flows (a real one is more than 1000 flows) because of the well-known combinatorial explosion problem. In [2], the study is extended by considering the scheduling of the flows in the network. This drastically reduces the number of scenarios.

In SpaceWire, the transmission of command/control messages needs to verify that messages can be delivered before their deadline. In [6], the computation of an upper bound of the worst case ETE delay of each message is proposed. Two methods have been studied: one based on Network Calculus and one based on a recursive flow analysis. These methods can analyze a complete Spacewire architecture but are pessimistic when very small packets are transmitted and when the traffic includes crossed flows. An industrial case study composed of 20 periodic flows sharing 4 Spacewire routers is also presented. The architecture seems to be small enough to be analyzed using timed automata theory. This paper proposes to model a Spacewire architecture in timed automata and to compute the exact worst case ETE delays using model-checking.

In Section 2, the behavior of a Spacewire architecture is encoded into timed automata. Section 3 is dedicated to the description of a case study. In this section, the worst-case delay analysis is computed and a complete scenario leading to this worst-case delay is proposed. A comparison between the method used in this paper and Network Calculus and Recursive Analysis is given in Section 4. Finally, Section 5 provides a conclusion and further work.

2 Modeling a Spacewire architecture using timed automata

This section proposes to model a Spacewire network architecture in timed automata and explains how to compute the worst-case ETE delays using model-checking.

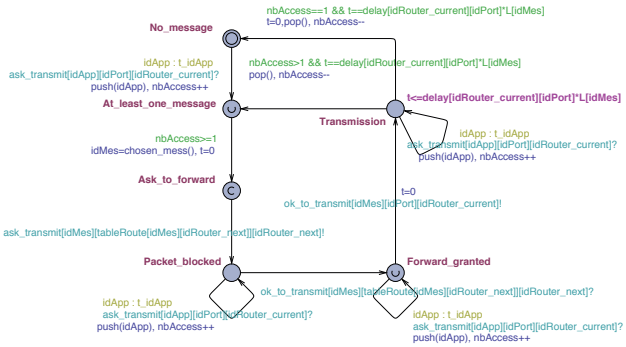


Figure 1. Automaton of a router output port

2.1 Timed automata overview

Timed automata have been first proposed by Alur and Dill [3] in order to describe systems behavior with time. A timed automaton is a finite automaton with a set of clocks, *i.e.* real and positive variables increasing uniformly with time. Transitions are labelled by a guard (condition on clock values), actions and updates which assign new value to clocks. Performing transitions requires no time. Conversely, time elapses in nodes. Each node is labelled by an invariant, that is a boolean condition on clocks. The node occupation is dependent of this invariant: the node is occupied if the invariant is true.

The composition of timed automata is obtained by a synchronous product. Each action a executed by a first timed automaton corresponds to an action with the same name a executed in parallel by a second timed automaton. The two transitions are performed simultaneously. Thus communication uses the rendez-vous mechanism.

Several extensions of timed automata have been proposed. The approach that is considered in this paper is based on timed automata with shared integer variables. The values of these variables can be consulted and updated by the different timed automata [7].

The modeling of a Spacewire architecture with timed automata is now presented. It is based on UPPAAL [7].

2.2 Modeling a Spacewire architecture

A Spacewire architecture is composed of periodic functions and routers. The timed automata system is then composed of one automaton per periodic function, which generates periodically a packet and one automaton per router output port, which models the transmission of packets on the output link, considering the blocking mechanism, the capacity of the link and the length of the message.

Figure 1 represents the timed automata model of an output port. When a packet is received by the output port, it is pushed in an input queue corresponding to its priority level. Then, the modeled behavior is as follows:

1. when the output port is free, a packet is chosen considering the wormhole routing policy. The output port of the router is then blocked for other packets;
2. the system immediately asks to transmit the packet to the next router. This simulates the transmission of the head of the packet to the next router;
3. while the signal *ok_to_transmit* is not received by the automaton, the packet is blocked. In the next router, three cases are possible: (1) the output port is free and the considered packet is chosen, the router sends the signal *ok_to_transmit* and the packet is released; (2) the output port becomes free and another packet is chosen, the considered packet is still blocked in all the upstream routers; and (3) the output port is waiting for the signal *ok_to_transmit* from a downstream router. So, the packet is blocked in the router and all its upstream routers. This behavior is generalized for all the routers and simulates the progress of the packet item by item in the network;
4. finally, when receiving the signal *ok_to_transmit*, the path to the destination is free and the packet is transmitted. The automaton waits for a transmission duration corresponding to the length of the packet (*L[idMes]*) times the capacity of the output link (*delay[idRouter][idPort]*).

The global model is obtained by combining both timed automata representing output ports of the routers, and timed automata modeling periodic functions.

Finally, the worst-case ETE transmission delays can be computed using the model-checking approach.

2.3 Computing the worst-case ETE delay

A system modeled with timed automata can be verified using a reachability analysis which is performed by model-checking. It consists in encoding each property in terms of the reachability of a given node of one automaton in the system. So, a property is verified by the reachability of the associated node if and only if this node is reachable from an initial configuration.

The worst-case ETE delay is obtained by verifying that all the packets are received before a bounded delay. This property is encoded by the test automaton depicted in Figure 2. When sending a packet, applications send immediately a signal *start_transmission*, which indicates the beginning of the transmission. The signal *end_transmission* needs to be received before a delay *bound* started when the test automaton receives the signal *start_transmission*. If not, the rejected node *bound_exceeded* is reached and the property is false.

We will now use the worst-case ETE transmission delays analysis presented here on a Spacewire case study.

3 Spacewire Case study

The case study of Figure 3 is a simplification of the one proposed in [6]. It is composed of application nodes A_i, a processor module PM and a mass memory unit MM. Applications are sensors or actuators. They receive commands (CMD) from the processor module and send back

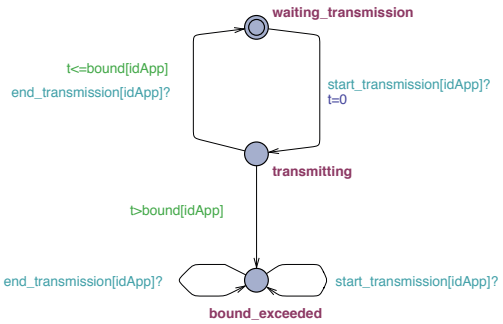


Figure 2. ETE delay test automaton

Traffic type	Path	Packet size (bytes)	Period (ms)
SC (Scientific)	$A_i \rightarrow$ MM	4000	20
HK (HouseKeeping)	$A_i \rightarrow$ PM	2000	4
CMD (Command)	PM $\rightarrow A_i$	1000	2

Table 1. Case study configuration

scientific data (SC) which are stored in the mass memory unit. They also send monitoring messages, named HouseKeeping (HK) messages, to the processor module. They also send monitoring messages to the processor module. Therefore, the network traffic is composed of 3 categories. Table 1 gives the network path and the size of the transmitted packets. All the flows are periodic.

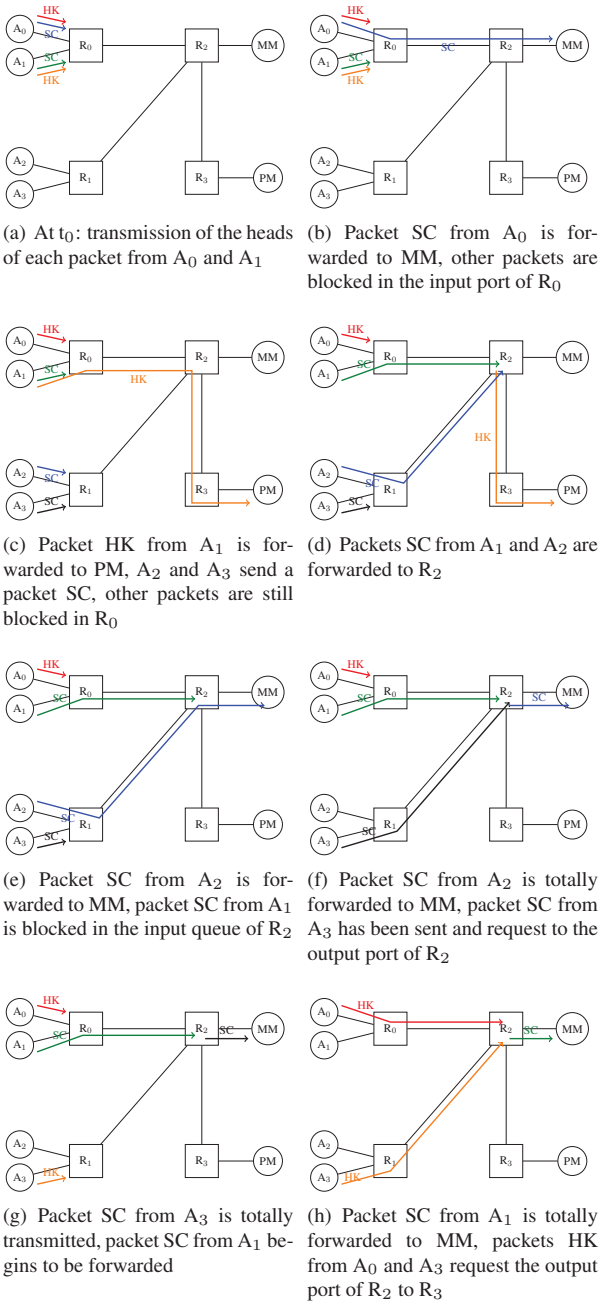
When the output port of a router becomes free, a round-robin procedure is used in order to choose which input port has to be selected. For this configuration, the order is A_0 then A_1 for router R_0, A_2 then A_3 for router R_1, and R_0 then R_1 for router R_2. The considered architecture is modeled in timed automata and is composed of two timed automata per A_i application, one sending a packet SC and one sending a packet HK, three timed automata per R_i which model the output ports of the routers, and one timed automaton which sends CMD packets from PM.

Using UPPAAL model-checker, we compute the worst-case ETE delays of each application flow. As an example, the worst-case ETE transmission delay of the housekeeping packets from A_0 takes 4.6ms and Figures 3(a) to 3(h) show a possible scenario which leads to this delay.

In the following section, we will compare the worst case ETE delay computed by the model-checking method and the one computed by Network Calculus and the Recursive Analysis.

4 Comparison with Network Calculus and Recursive Analysis

In [6], Network Calculus (NC) and Recursive Analysis (RA) are applied to the worst-case ETE transmission delays on an industrial case study. These solutions are pessimistic in some situations: RA cannot handle very small packets correctly, especially if a bottleneck is present such as a slow terminal and NC has trouble when the traffic includes crossed-flows. Conversely, model-checking (MC) always give the exact worst-case ETE transmission delays.



(a) At t_0: transmission of the heads of each packet from A_0 and A_1

(b) Packet SC from A_0 is forwarded to MM, other packets are blocked in the input port of R_0

(c) Packet HK from A_1 is forwarded to PM, A_2 and A_3 send a packet SC, other packets are still blocked in R_0

(d) Packets SC from A_1 and A_2 are forwarded to R_2

(e) Packet SC from A_2 is forwarded to MM, packet SC from A_1 is blocked in the input queue of R_2

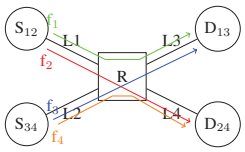
(f) Packet SC from A_2 is totally forwarded to MM, packet SC from A_3 has been sent and request to the output port of R_2

(g) Packet SC from A_3 is totally transmitted, packet SC from A_1 begins to be forwarded

(h) Packet SC from A_1 is totally forwarded to MM, packets HK from A_0 and A_3 request the output port of R_2 to R_3

Figure 3. Worst-case ETE delay scenario for A_0 HouseKeeping packet

As an example, the network architecture of Figure 4 allows to compare the three methods. It is composed of 4 applications and a router [6]. The bound of the worst-case ETE delays is computed considering the crossed paths and by varying the capacity of the link L_4. Table 4(b) shows the configuration of different studied scenarios and gives the computed ETE delays in ms of the different methods. In the first scenario, the RA gives the optimal bound but not the NC. And in the second scenario,both f_2 and f_4 sends small sized packets. NC gives better results than



(a) Architecture

Flow	Scenario 1 (L_4 = 50 Mbps)					Scenario 2 (L_4 = 0.2 Mbps)				
	Size	Period	NC	RA	MC	Size	Period	NC	RA	MC
f_1	4000	20	2.08	1.99	1.98	4000	20	3.8	10	2.8
f_2	500	8	2.26	1.99	1.98	20	32	4.6	16.2	3.8
f_3	5000	20	2.06	1.99	1.98	5000	20	3.8	10	2.8
f_4	400	8	2.06	1.99	1.98	20	32	3.8	16.2	3.8

(b) Configuration and results

Figure 4. Case study including bottleneck and crossed flows

the RA. For the two scenarios, model-checking gives the exact worst-case ETE transmission delays. They are close to those computed by the RA in the first scenario. The difference is due to the numeric approximations of the methods. The pessimism of the Recursive Analysis and Network Calculus can be determined for the second scenario.

5 Conclusion and further work

The paper proposes a model-checking approach to compute the exact worst-case ETE delays of Spacewire periodic flows. In Spacewire architecture, wormhole routing is used to share communications on the network. This mechanism has been modeled using timed automata theory. A Spacewire case-study is proposed. Its configuration is composed of 9 flows and 4 routers and is smaller than a realistic configuration composed of at least 20 flows. The computation of worst case ETE delays for the case study shown in this paper takes more than 1h on a Macbook with 2.2 GHz Intel Core i7 processor having 8 GB RAM. However, by adding only one application, the evaluation cannot be performed in reasonable amount of time. This result is not surprising as model-checking executes an exhaustive analysis of all the scenarios. Therefore, two problems have to be considered: the scale of time units and the number of packets in the network.

In one hand, in a Spacewire configuration, the period of the applications is a few milliseconds. And the transmission delays takes a few nanoseconds. The model-checking procedure considers all the possible valuations of the clocks which leads to a huge number of states.In another hand, due to the transmission delays of the packets and the periods of the applications, several packets of an application can be in the network and have to be taken into account when computing the worst-case ETE delays of the packets they influence, *i.e.* packets which share the same path, and lead to a huge number of scenarios.

In the context of the AFDX networks, the worst-case ETE delay occurs when the waiting time in the output ports of the switches is maximized. Thus, the problem is to find a scenario which maximizes this waiting time. A scenario is defined by the sequences of messages generated by the different applications and by the instant of the first message sent by the application. Messages are characterized by a Bandwidth Allocation Gap (BAG). The BAG parameter is the minimum delay between two consecutive message transmissions. So, for each application, it is possible to construct a periodic sequence of messages by considering the BAG parameter. A small number of configurations are worst-case scenario candidates: scenarios where, at each switch output port, the message under study arrives at the same time as one message from all other input links of the corresponding switch output port. In [2], the timed automata modeling takes into account the real scheduling of the packets. Thus, thanks to the BAG parameter, the number of possible scenarios can be reduced and AFDX networks with up to 32 flows can be analyzed. In a Spacewire architecture, because there is no temporal relationship between the messages, the method used in the AFDX context cannot be directly applied. The problem is to build, for each Spacewire application, all the sequences of messages which are candidate for the worst-case ETE delay analysis. Further studies are needed to optimize the method in order to analyze an industrial size Spacewire architecture.

References

[1] IEEE Standard for Heterogeneous Interconnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction). *IEEE Std 1355-1995*, 1996.

[2] M. Adnan, J. Scharbarg, J. Ermont, and C. Fraboul. An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows. In *Proc. of 17th Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, September 2012.

[3] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach. *IEEE Transactions on Industrial Informatics*, 6(4):521 –533, nov. 2010.

[5] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an AFDX network. In *Proc. of 18th Euromicro Conference on Real-Time Systems*, pages 193–202, 2006.

[6] T. Ferrandiz, F. Frances, and C. Fraboul. A Sensitivity Analysis of Two Worst-Case Delay Computation Methods for SpaceWire Networks. In *Proc of 24th Euromicro Conference on Real-Time Systems*, July 2012.

[7] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[8] M. Lauer, J. Ermont, C. Pagetti, and F. Boniol. Analyzing End-to-End Functional Delays on an IMA Platform. In *ISoLA*, volume 6415 of *LNCS*, pages 243–257. Springer, 2010.

[9] S. Parkes and P. Armbruster. SpaceWire: a spacecraft onboard network for real-time communications. In *Real Time Conference, 2005. 14th IEEE-NPSS*, pages 6 –10, june 2005.